Parallel Unstructured AMR and Gigabit Networking for Beowulf-Class Clusters

Charles D. Norton and Thomas A. Cwik

Jet Propulsion Laboratory
California Institute of Technology
MS 169-315, 4800 Oak Grove Drive, Pasadena CA 91109-8099, USA
{Charles.D.Norton, Thomas.A.Cwik}@jpl.nasa.gov
http://hpc.jpl.nasa.gov/APPS/AMR

Abstract. The impact of Gigabit networking with Myrinet 2000 hardware and MPICH-GM software on a 2-way SMP Beowulf-Class cluster for parallel unstructured adaptive mesh refinement using the *PYRAMID* library is described. Network performance measurements show that approximately 225 Mbytes/s (1.8 Gbits/s) with 9.3μ seconds latency can be achieved for large messages in ping-pong tests. The performance varies depending on how processors communicate; either within an SMP node or across nodes. When applied to parallel unstructured AMR of 3D tetrahedral meshes noticeable improvement is observed when compared to 100BaseT (100 Mbits/s) Ethernet. Furthermore, when compared to traditional systems, such as the SGI Origin 2000, one realizes that Beowulf-Clusters with fast networks and high performance SMP processors compare favorably with such systems–even for communication intensive applications.

1 Introduction

Beowulf-Class clusters have demonstrated that scalable parallel computing can be achieved, at low cost, through the use of commodity off-the-shelf (COTS) parts. The COTS approach allows one to configure a system using the latest hardware and software that is available, and affordable, at the time. As time goes by, one has the option to reconfigure a system as new products are announced and released. Many cluster components may be freely available, such as system software, while others, including advanced microprocessors, are constantly under competitive pricing pressures to remain affordable.

The network interconnect, however, allows great flexibility regarding the communication performance one expects versus the amount one is willing to spend. Although 100BaseT Ethernet cards are not expensive the 100 Mbits/s (theoretical) upper bound can hinder the performance of clusters containing many fast SMP processors with large amounts of main memory. Nowadays, improvements in system software and microprocessor performance bring added pressure to consider fast networking to maintain a balanced system.

Our most powerful cluster contains 31 compute nodes, plus one front end node, of dual-processor 800 MHz Pentium III's–a total of 64 processors in all. Each node has 2 GBytes of RAM available giving a system with 128 GBytes of main memory with 51.2 GFlops of computation peak. We added Myricom's new Myrinet 2000 networking hardware to our existing Cisco 100BaseT Ethernet switch. More details about Myricom's technology including architecture specifications, software, algorithms, and products are available at their web site and elsewhere [1, 6]. The availability of two networks allows us to evaluate the impact of Gigabit networking for our system. We will also draw comparisons to our nearest competitor–a 128 cpu SGI Origin 2000 with 64 GBytes main memory, 76.8 GFlops peak computation (2 flops per clock) with a custom NUMA architecture network.

2 Characterizing Communication-Intensive Applications

Many physics-based numerical applications are fundamentally irregular, meaning that the solution process is largely determined at run-time and the communication requirements are non-uniform, although generally predictable. Parallel adaptive methods fall into this category and software for these techniques require very high performance systems for large problems.

Our *PYRAMID* parallel unstructured AMR library supports finite element applications [5, 7]. Large triangular and tetrahedral meshes can be loaded, adaptively refined with automatic mesh quality control, load balanced, and migrated among the processors using high level object-based library commands.

Many of these routines accept Fortran 90/95 optional arguments that allow for additional control over the actions taken. When these options are not specified a reasonable default action is taken. Furthermore, for the most sophisticated routines, keyword arguments may optionally be specified to visually remind the user how various arguments are used.

The new features of Fortran 90/95 allow for complex data structures to be created, and used, while supporting encapsulation of related concepts in modules. Use of modules make the data and routines they contain available to program units and they can interact to provide increased functionality. Features such as smart pointers, generic interfaces, whole and array subset operations, and dynamic storage provide the modern software principles required for this work. This Fortran-based approach will simplify the union of adaptive meshing with existing Fortran-based solvers.

Since parallel adaptive mesh refinement potentially involves working with many millions of elements great effort is used to minimize communication and to ensure that transmitted messages are as large as possible. On a cluster, managing communication becomes even more important since applications may use networks that are significantly slower than those found on most traditional supercomputing systems.

Figure 1 shows a segment of a large tetrahedral artery blood flow mesh segment. The original geometry was provided by Taylor et. al [8] and the initial

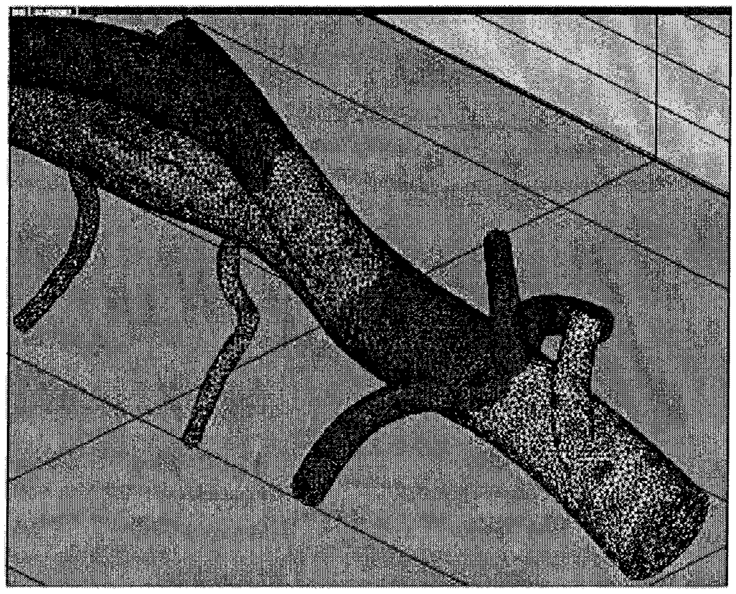


Fig. 1. Repartitioning and migration of artery mesh segment using PYRAMID.

mesh was generated by the Scientific Computation Research Center at Rensselaer Polytechnic Institute [2]. The mesh contains 1.1 million elements where our *PYRAMID* AMR library was used for repartitioning, load balancing, and mesh migration as illustrated.

Mesh migration performance comparisons for this problem, using 100BaseT Ethernet and Myrinet on the cluster as well as the NUMA architecture of the SGI O2K, indicate the benefit of Myrinet as shown in figure 2. This is largely a communication-based benchmark. As we will see, however, the performance tradeoffs vary based on the problem solved. Nevertheless, this initial benchmark indicates that the Beowulf cluster can compete with traditional systems even for communication intensive applications.

3 Myrinet, Fast Ethernet, and the SGI Origin Network

Although cluster technology has generally stabilized integrating new components into an existing system can cause problems. At this writing, our Beowulf cluster is based on Redhat Linux 6.2 with Kernel version 2.4.19 SMP using MPICH-GM version 1.2.1..7 and GM-1.5 as the low level communication subsystem. Previous versions of MPICH-GM had problems, but the most recent release has good stability with improved performance. Both the GM layer and Myricom's version of MPICH, called MPICH-GM, have been improved. Our hardware consists of the Myrinet M3E32 Switch, PCI 64B, Lanai 9 with 4 MB SDRAM. Our Super-

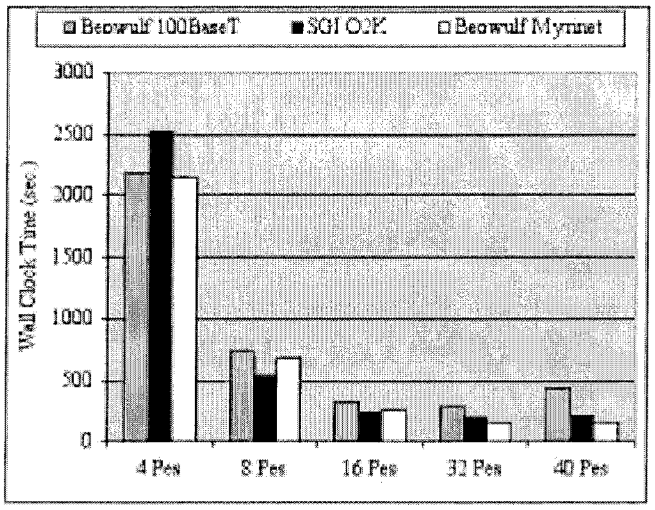


Fig. 2. Performance for communication bound mesh loading, repartitioning, and migration of the artery mesh.

Micro SUPER 370 DLE Motherboards use the ServerWorks ServerSet LE rev 5 chipset.

Early versions of MPICH-GM required users to specify a number of performance related parameters regarding memory registration, shared-memory support, and so on. More recent versions hide these details and have other useful features, such as the capability to decide at run time if shared-memory communication should be used. The ability to have shared-memory communication as an option was important for PCI chipsets that exhibited poor performance since this option often improved performance. For our system, the peak PCI bandwidth is 455 MBytes/s which is very good. On older systems this was often a "hidden" bottleneck to network performance since the PCI could be as low as 125 MBytes/s. We experience better performance for large messages when shared-memory is not used.

Figure 3 shows the results of a network ping-pong test on two processors for the Myrinet installation compared to previous results for 100BaseT and the SGI Origin 2000. Incidentally, the MPI implementation on the Origin uses global shared memory to implement message passing. When a processor requires data the packets are sent over the CrayLink so the latency to access memory becomes a critical part of a performance metric on this machine in addition to good cache management.

The improvement for our cluster is significant where neighbor processors that are not on the same board are used. This certainly had an impact on the artery mesh migration problem in figure 2. While we are very close to peak speed for 100BaseT we also achieve near peak speed for MPICH-GM on the Myrinet 2000

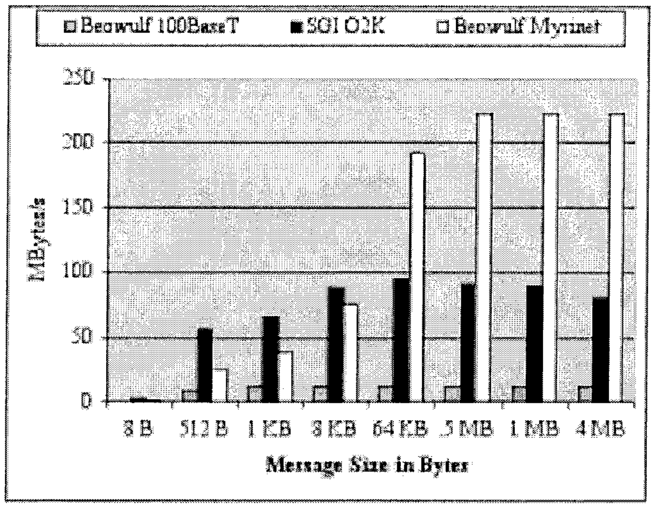


Fig. 3. MBytes/s transmitted in ping-pong tests between two processors over the network using the mpptest program. Results for messages above 4 MB are fairly constant.

hardware (rated at 2 Gbits/s) for ping-pong tests. The average latency is still quite low, about 1.5μ sec. for processors that share a CPU board and 9.3μ sec. for processors on separate CPU boards. The Myrinet result in figure 3 uses processors on separate boards so shared-memory communication is not an issue.

4 Evaluating Functionality and Performance

Prior to examining how the Myrinet 2000 hardware performs on our adaptive meshing simulations we should take a closer look at network performance in an SMP environment.

The MPICH-GM results in figure 4 show that good performance is possible, particularly for large messages. There is a cross-over point where communication between processors on the same CPU board (node) falls behind processor communication across boards for messages between 32-256 KBytes. This is primarily due to cache effects since the receiving processor can access nearby data directly from the cache, instead of from the shared-memory region, but only up to a point. Nevertheless, for our adaptive meshing problem we regularly send messages as large as 35 MBytes in size and we do not use shared-memory communication in MPICH-GM so our simulations will select processors one at a time, one board at a time.

4.1 Analyzing the Muzzle-Brake Mesh

Figure 5 shows a muzzle-brake shock tube mesh with its initial partitioning and redistribution among processors. We repeated the simulations from [7] using

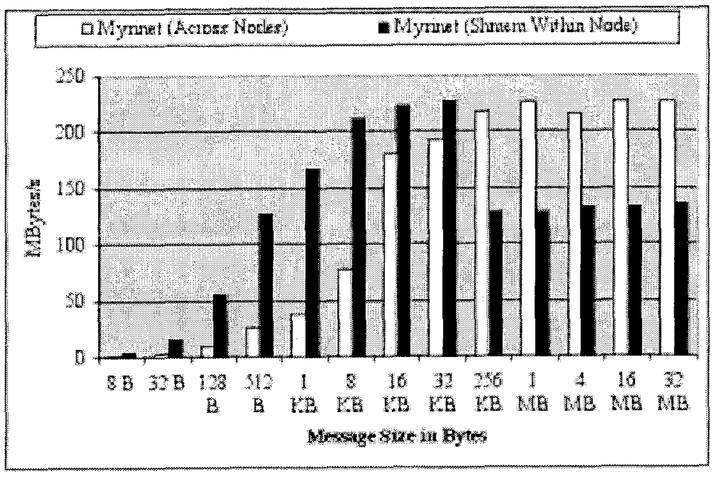
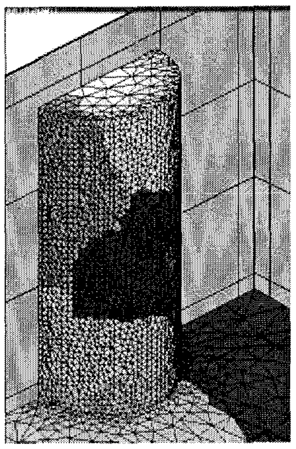


Fig. 4. MBytes/s transmitted in ping-pong tests between two processors on the same node (using shared-memory communication) and across nodes using MPICH-GM.



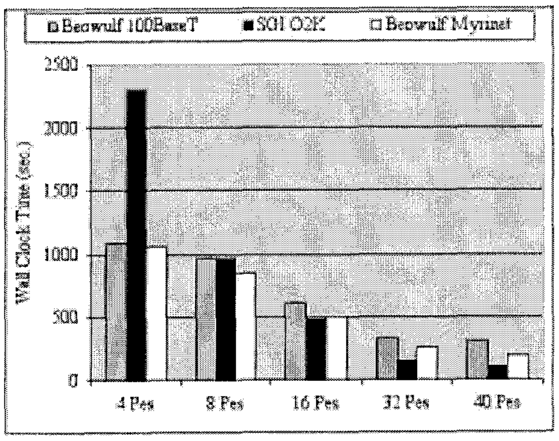


Fig. 5. Muzzle-brake shock tube mesh with initial mesh partitioning and redistribution among eight processors. Performance results after three adaptive refinements of the shaft section are also included.

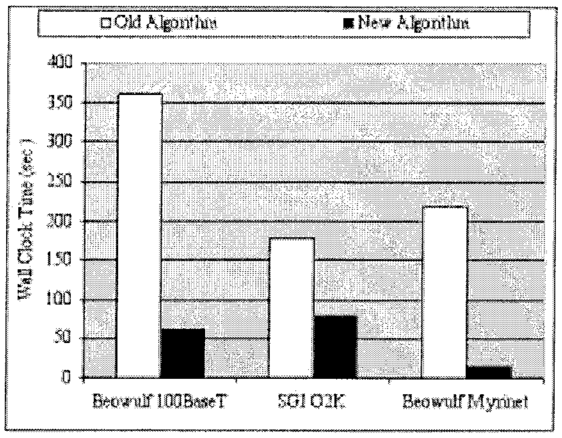


Fig. 6. Migration stand-alone timing for communication dominate Earthquake mesh generation on 8 processors due to message passing algorithm improvements. Various performance comparisons can be derived.

Myrinet 2000 and have included the combined results in figure 5. The initial mesh contains just 34,214 elements, but after three adaptive refinements it contains 1,264,443 elements.

What is clear is that for a small number of processors the Beowulf cluster performs much better than the Origin for this problem. As the number of processors is increased all configurations show some scalability, but the Origin outperforms the cluster. One would suspect that a fast Myrinet network would show an even greater improvement over the 100BaseT Ethernet. This is misleading, however, because the communication performance is not dominant for this mesh.

A breakdown of the time spent in mesh migration, which includes partitioning and load balancing, compared to creating new elements by adaptive refinement shows that much more time is spent in the AMR process. On the Beowulf cluster using Myrinet ~182s and ~46s are spent in creating new elements and migrating them respectively. Similarly, ~98s and ~22s are respectively spent in these stages on the Origin. In fact, once the coarse elements have been redistributed the new elements are created locally so no communication is required. This implies that improvements in the network will not significantly impact overall performance for this problem instance.

4.2 Network Hardware Inspired Algorithm Modifications

Improvements in algorithm design for communication on clusters should also be attacked given fast networking. Figure 6 is one example where the migration time is measured for 8 processors based on old and new communication algorithm techniques on an earthquake mesh. The original algorithms set up

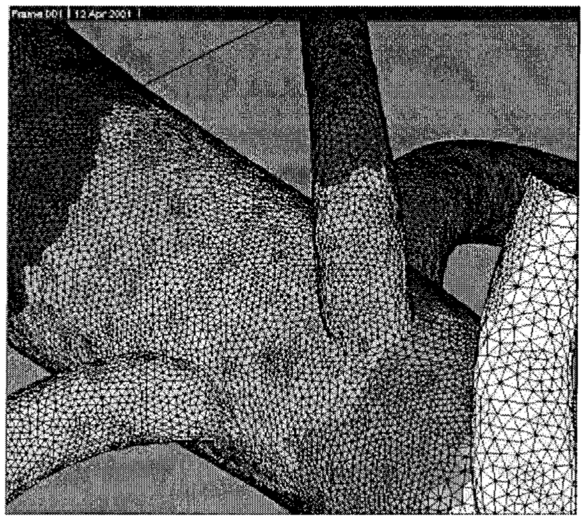


Fig. 7. Adaptive refinement, repartitioning, and migration of the artery mesh segment containing 1.8 million elements.

communication schedules for irregular data movement by sending and receiving messages directly as needed. That approach assumed that good performance can be achieved by matching the minimum number of communication operations exactly.

The new communication algorithm is much more scalable for large systems in that a tree-based pair-wise exchange algorithm is used, reminiscent of early hypercube routing algorithms. This algorithm works on any number of processors. It also guarantees that a minimal number of exchanges will be performed–meaning that the algorithm can determine if a processor has already received the data it needs and skip communication operations as necessary. The volume of data tends to grow with such algorithms as exchanges are performed, but this algorithm can also determine when data need not be included in future stages removing it from such operations. It is designed to take advantage of networks supporting full-duplex communication and multiple routing paths such as Myrinet.

This approach also works well for clusters with 100BaseT Ethernet, as seen in figure 6. When Myrinet is applied the results are even more dramatic. All of the performance results in this paper are based on using the new communication algorithms.

4.3 Analyzing the Artery Mesh

Returning to our artery mesh figure 7 shows the adaptive refinement of a small section that creates 1.8 million elements from the initial mesh of 1.1 million elements. Figure 8 shows again how performance can vary between the Myrinet-based Beowulf cluster and the SGI Origin 2000.

The first instance in figure 8 adaptively refined a small region and analysis showed that the time spent in migration and adaptive refinement were nearly equal across both machines. In the second instance the region refined was increased, creating about 4.5 million elements. For the cluster both the migration and adaptive refinement time increased significantly, but more time was spent in adaptive refinement than migration. This pattern was also true for the SGI Origin, but both migration and refinement were faster than on the cluster.

In the third instance the entire artery was refined creating 7.4 million elements. In this case, where one would expect the cluster to perform poorly, it actually outperformed the SGI Origin. Analysis showed that the migration time for the cluster was slower than for the Origin while the adaptive meshing time was faster for the cluster than for the Origin. Overall, this gave the cluster better performance in this case.

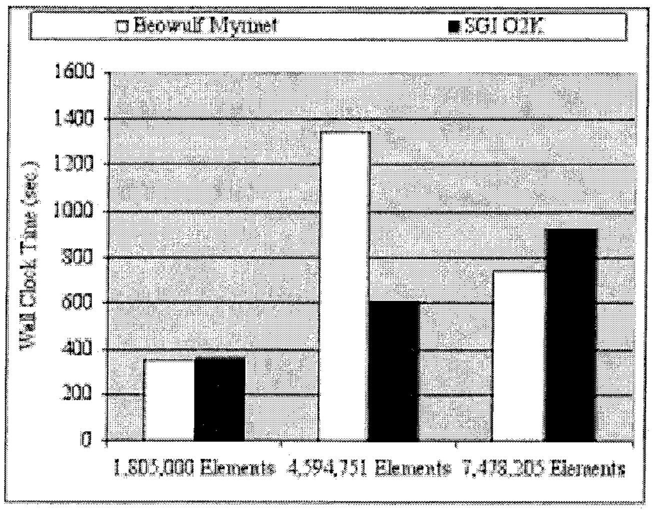


Fig. 8. Performance comparison for adaptive refinement of artery mesh for various region sizes on 50 processors.

We also examined the message passing structure more closely for these problem instances. Although the new pair-wise message exchange algorithm was used for mesh migration we decided to analyze the mapping for processor communication exactly. This included measuring, for each processor, the number of other processors that need to send data as well as the number of elements each processor must receive in order to achieve load balance during migration. Neither of these measurements allowed specific conclusions to be drawn about the performance differences among these problems cases. In fact, the data interaction was fairly well balanced for all problem sizes under both architectures. There were instances, however, where the number of elements a small number of processors must receive for load balance was about 5 times larger than the average, but this was characteristic of all test cases. This suggests that characteristics of the problem may determine performance much more than features of the network, and that for irregular problems this is difficult to characterize.

5 Conclusion

As one would expect, the upgrade of a network mainly benefits problems with large message passing requirements. On point-to-point ping-pong tests using Myricom's MPICH under the GM messaging layer we can achieve near peak performance, with low latency, for sufficiently large messages. On SMPs where shared-memory communication is possible, we also observed a crossover where communication on the dual-CPU board falls behind communication across boards for large messages. Ultimately the ability to sustain network performance for large messages is more important for users running large applications.

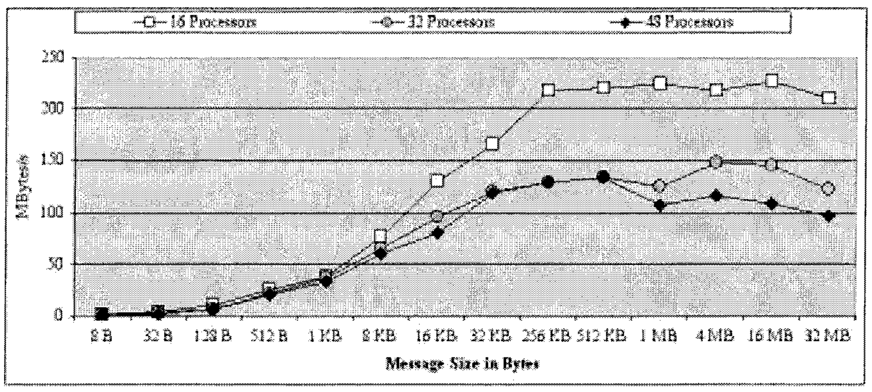


Fig. 9. Bisection bandwidth measurements by processor where the aggregate bandwidth is normalized by the bisection partition size.

Figure 9 shows a measurement of the bisection bandwidth for various message sizes across multiple numbers of processors. The data has been normalized by the number of processors in a partition. A range of performance variations can be seen based on the number of processors used in the simultaneous communication. For large messages the performance is good for a small subset of processors, but as the number of processors increases (implying more traffic on the network), the performance drops. This is likely a contributing factor to the performance of our adaptive meshing application which performs exchanges for large messages. In these tests shared-memory communication is not applied.

One caveat regarding performance comparisons between the Beowulf SMP Cluster and the SGI Origin 2000 for our adaptive mesh problem is that the simulation results are not precisely identical. In particular, the ParMetis partitioner applied in the dynamic load balancing stage produces different partitionings on each machine [4]. (This is a side-effect of the random numbers used in the partitioning algorithms.) This will affect the adaptive refinement process and may affect comparative timings, but it should be not too significant.

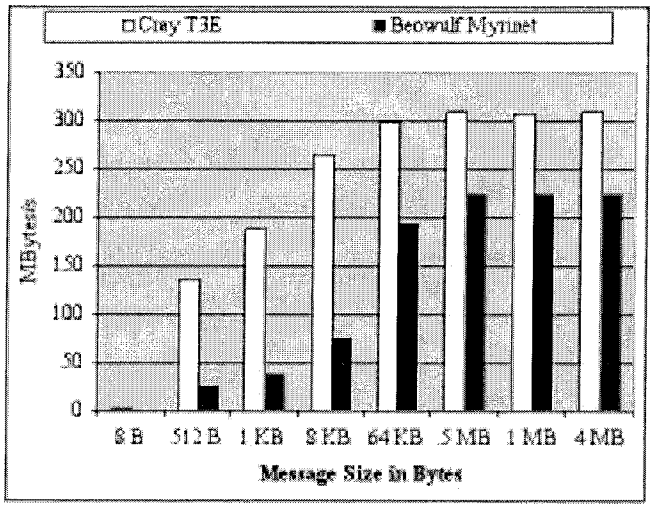


Fig. 10. Performance comparison between Cray T3E and Pentium-III Beowulf cluster for ping-pong tests between two processors in MBytes/s.

Finally, it is interesting to compare how well commodity networks, such as Myrinet, compare to highly rated machine specific networks such as the Cray T3E. Figure 10 gives this comparison which is reasonably good for a commodity cluster with Myrinet 2000. Although the Cray T3E bandwidth is higher the latency is also higher measured at 34.71μ sec. compared to 9.3μ sec. for Myrinet.

Installing updated MPICH-GM software had a large impact on our system reliability and stability. In the end, the network is just one contributor to the combination of factors that affect performance and usability of the cluster. Our experience is that good network performance can be achieved using Myrinet 2000, but the effective benefit for applications depends largely on their characteristics.

Acknowledgement We appreciate several helpful discussions with members of the JPL High Performance Computing Group, JPL Supercomputing Group, and the Myricom Technical Support team. This work was performed at the Jet

Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration.

References

1. N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet - A Gigabit-per-Second Local-Area Network. *IEEE Micro*, 15(1):29–36, February 1995.
2. Joseph E. Flaherty and James D. Teresco. Software for Parallel Adaptive Computation. In Michel Deville and Robert Owens, editors, *Proc. 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, Lausanne, 2000. IMACS. Paper 174-6.
3. J. Hsieh, T. Leng, V. Mashayekhi, and R. Rooholamini. Architectural and Performance Evaluation of GigaNet and Myrinet Interconnects on Clusters of Small-Scale SMP Servers. In *Proc. SC'2000*, Dallas, Texas, November 04–10 2000. IEEE Computer Society. CD-ROM.
4. G. Karypis, K. Schloegel, and V. Kumar. ParMetis: Parallel Graph Partitioning and Sparse Matrix Ordering Library. Technical report, Dept. of Computer Science, U. Minnesota, 1997.
5. J. Z. Lou, C. D. Norton, and T. Cwik. A Robust Parallel Adaptive Mesh Refinement Software Package for Unstructured Meshes. In *Proc. Fifth Intl. Symp. on Solving Irregularly Structured Problems in Parallel*, 1998.
6. Myricom, Inc. *Myricom Creators of Myrinet*, 2001. http://www.myri.com.
7. C. D. Norton, J. Z. Lou, and T. A. Cwik. Status and Directions for the PYRAMID Parallel Unstructured AMR Library. In *15th International Parallel and Distributed Processing Symposium*, San Francisco, CA, April 23-27 2001. Irregular 2001 Workshop. CD-ROM.
8. Charles A. Taylor, Thomas J. R. Hugues, and Christopher K. Zairns. Finite Element Modeling of Blood Flow in Arteries. To appear, *Comp. Meth. in Appl. Mech. and Engng.*, 1999.